

# FEATURE FUSION FOR STOREFRONT RECOGNITION AND INDOOR NAVIGATION

Ziwei Xu<sup>1</sup>, Minjian Pang<sup>2</sup>, Guyue Zhou<sup>3</sup>, Lu Fang<sup>2</sup>

University of Science and Technology of China<sup>1</sup>  
 Hong Kong University of Science and Technology<sup>2</sup>  
 Dajiang Innovations Co., Ltd.<sup>3</sup>

## ABSTRACT

With the help of a map and GPS, outdoor navigation from one spot to another can be done quickly and well. Unfortunately inside a shopping mall, where GPS signal is hardly available, navigation becomes troublesome. In this paper, we propose an indoor navigation system to address the problem. We propose a DNN-based feature fusion method to help automatically identifying shops in a photo. During navigation, we make use of this method to find out the realtime position of the user. We utilize multiple image processing techniques to parse photos of a mall's shopping instruction and a construct topological map of the mall. Unlike most existing indoor navigation systems, which relies heavily on infrastructures and pre-labelled maps, our system uses only photos taken by cell-phone cameras as input.

**Index Terms**— Indoor Navigation, Scene Recognition, Feature Fusion, Topological Map

## 1. INTRODUCTION

Indoor positioning system (IPS) has been a research highlight in recent years. Methods based on radio waves, magnetic fields, acoustic signals, or other sensory information collected by mobile devices have emerged [1]. These methods relies heavily on pre-installed infrastructures or special receivers, for example, RFID transmitters/receivers [2], configured fluorescent light receiver [3] or many Wi-Fi access points [4]. While these methods can achieve high precision in positioning, deploying such devices in a shopping mall can take much time and effort, and may not be user-friendly in real application.

In this article, we propose a new vision-based indoor positioning system that relies on no other infrastructures but a camera. To achieve this, document layout analysis (DLA) techniques, text detection and scene recognition methods are employed. DLA methods are used to extract information from the map provided by most shopping malls. Ngram-based text classification and scene recognition methods are used to recognize shop in realtime environment. However, we see some drawbacks in traditional way of using pure text information for shop recognition and put forward a new feature fusion

method for higher precision.

The system consists of four modules (see Figure 1). The first module is an image parser, where images of shopping instructions (floorplan and shop list) are parsed and topological map of the shopping mall is constructed. This module only functions when the user goes to a shopping mall and takes photos of the floor plan and shop list for the first time. Related content is in **Topological Map Construction** section. The second module is a realtime environment sensing module, where environment images are processed and matched with the topological map. The output of the former two modules is used by localization module. Finally, the localization result is used by navigation module, which provides path from the current position to a specified shop. The introduction for the latter three modules is in **Environment Sensing and Localization** section.

## 2. TOPOLOGICAL MAP CONSTRUCTION

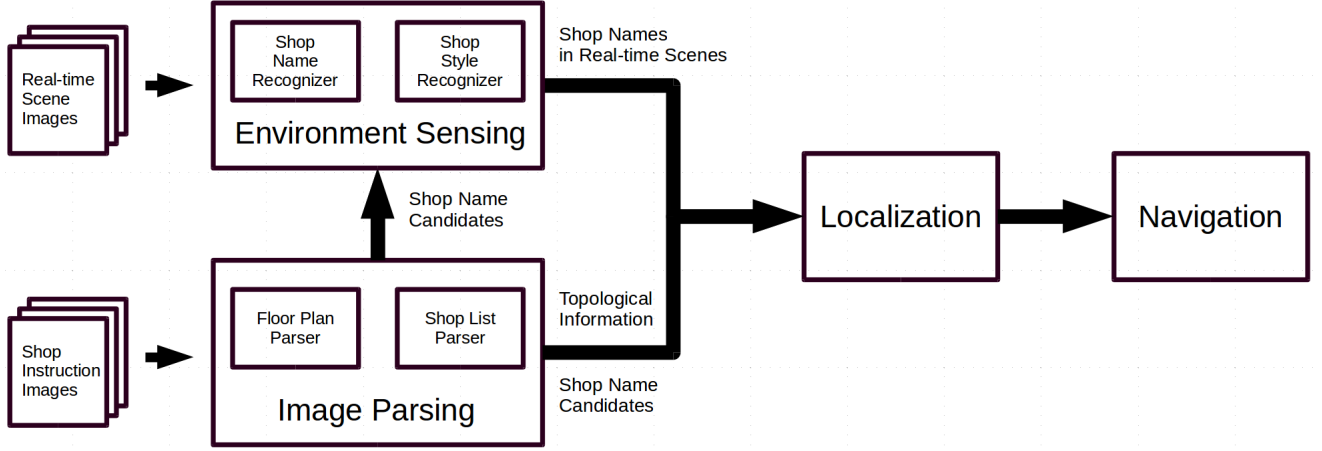
In this step, two images containing respectively a floorplan and a shop list are used. Multiple image processing and document layout analysis techniques are utilized in several steps so that information can be correctly extracted. The output of this module is a topological map represented by a graph. Each node of the map is an accessible area with a certain set of shop nearby.

### 2.1. Image Preprocessing

A shopping mall floorplan typically contains a set of icons, texts and colored blocks. Icons and texts provides semantic information about the map (i.e., name or ID of a shop), while colored blocks provide topological information. In preprocessing step, texts/icons are detected, then recognized and finally removed from map image so that topological information can be extracted correctly later.

#### 2.1.1. Text Detection and Recognition

In most shopping mall floorplans, there is a significant contrast between text elements and other components. Therefore, texts can be extracted by detecting maximally stable extremal



**Fig. 1.** System Architecture

regions (MSERs) [5]. Following the method described in [6], we detect MSERs in the image as CCs. Detected CCs are filtered based on their size, eccentricity and aspect ratio (i.e.,  $w/h$ , where  $w$  is the width of the bounding box and  $h$  the height). Filtered CCs are then clustered based on their relative distances to form a word.

For text recognition, we use the open source tesseract-ocr software package [7]. It performs OCR in a bottom-up manner, where CCs are first detected independently and then clustered together to form text blobs and lines. Finally those blobs and lines are recognized as characters or words.

Tesseract-ocr gives fairly good output in most formatted documents. However, in our experiment, it often fails when processing texts on a floorplan, missing most of the texts and gives wrong recognition results. In our practice, we locate texts using MSER-based method mentioned above. Then we crop text areas from the image and deliver them to tesseract software and get its recognition result. In this way, almost all texts on a floorplan can be recognized correctly.

### 2.1.2. Text Removal

Texts and icons are removed after being localized and recognized. We regard texts as corruptions and apply inpainting method introduced in [8] and [9] to smoothly remove them. Inpainting keeps color blocks in the image smooth so that later a segmentation operation can be done correctly to separate accessible area and different shops. For simplicity purposes, we will refer to the image with texts and icon removed as image  $R$ .

## 2.2. Floor Plan Parsing

In this step, accessible area and shop blocks are separated and labelled. Accessible area and shop blocks are separated first. After that, shop blocks are further segmented so that each seg-

ment corresponds to a shop. Finally the accessible area is segmented into different nodes. In each of the three operations, we use different techniques for best result.

### 2.2.1. Accessible Area Extraction

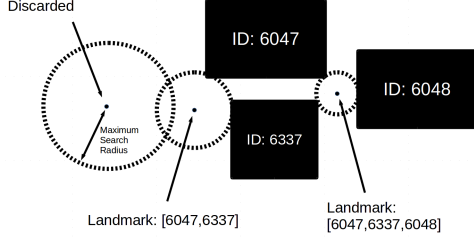
In a floorplan, accessible area (AA) usually has an intensity different from any component else so that viewers can easily find out where they can go. Based on this presumption, image binarization and CC analysis is used to identify accessible area. The image  $R$  is binarized based on Otsu's thresholding method as image  $B_1$ . The negative version of  $B_1$  is also obtained as  $B_2$  so that accessible area can be found no matter it is darker or brighter than shop blocks. On  $B_1$  and  $B_2$ , CC with a size smaller than 20 pixels are removed. After this, we define the CC corresponding to accessible area (denoted as  $CC_A$ ) as:

$$CC_A = \operatorname{argmax}(\frac{D(CC_i)}{\operatorname{Area}(CC_i)} - E(CC_i)), CC_i \in CC \quad (1)$$

where  $E(\cdot)$  denotes the Euler number [10],  $\operatorname{Area}(CC_i)$  denotes the area of the  $CC_i$ 's outer rectangle, and  $D(\cdot)$  is defined as:

$$D(CC_i) = \sum_{j \neq i} I_{CC_i}(CC_j), \quad (2)$$

where  $I(\cdot)$  is the indicative function. An Euler number is the number of CC object minus the number of holes in the object. On one hand, shop blocks are surrounded by accessible areas, resulting in many 'holes' in it and its Euler number being the minimum. On the other hand, accessible areas usually span the whole map, making its outer rectangle containing many CCs that corresponds to shop blocks. However, note that the CC that corresponds to the background also contains many CCs, therefore we divide  $D(CC_i)$  by its outer rectangle's area. In most cases, maximizing equation 1 gives us the accessible area component.



**Fig. 2.** Accessible area segmentation. A circle centered at an AA pixel with adaptive radius is formed and shop blocks inside the circle are added into landmark list of the pixel. Pixel that has no shop nearby will be discarded.

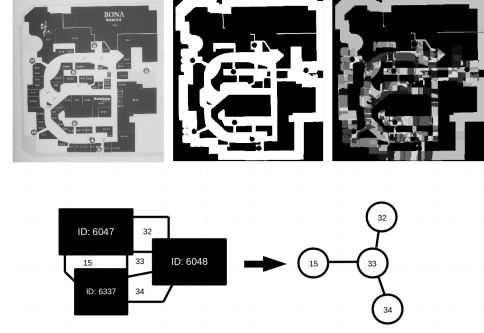
### 2.2.2. Shop Block Segmentation

The rest of CCs after accessible area extraction are considered as shop blocks. Each of these blocks corresponds to one or more shops. To further segment these CCs, we obtain the original pixels of each CC from image  $R$  and perform recursive flood-fill algorithm on them. The stopping condition for flood-fill is that the difference between current pixel and last pixel is greater than a threshold value. We choose 25 as the threshold in our experiment. This method is based on the presumption that different shops are separated by curves with distinguishable intensities.

After this step, every pixel on the floorplan is assigned an integer. Each integer corresponds to either a shop or an AA. Integers representing shops will then be mapped with the texts extracted in the before steps. This mapping is performed based on the position of the text label. Shop block that has no text label inside will be considered as obstacles and will be discarded.

### 2.2.3. Accessible Area Segmentation

In order to perform navigation, the accessible area needs to be further segmented based on the different attributes of its pixels. In our system, we treat each pixel of the AA as an observation spot. For each spot, the algorithm search the neighborhood for labelled shop blocks as landmarks. The range for landmark searching is adaptive such that each spot has at most 4 landmarks being stored. AA pixels with no shop blocks nearby are discarded. (See Figure 2) Then pixels (spots) with same landmarks are grouped together to form a node. An undirected weighted graph is constructed to represent the topological layout of the floorplan. The weight of an edge is the distance between centroids of two nodes on the ends of the edge. Figure 3 shows the result of AA segmentation and how the graph is constructed.



**Fig. 3.** Accessible area segmentation and graph construction. Accessible area are segmented based on the "view" of each AA pixels. Segmented accessible areas are treated as nodes and adjacent nodes are linked by edges to form a graph.

### 2.3. Shop List Parsing

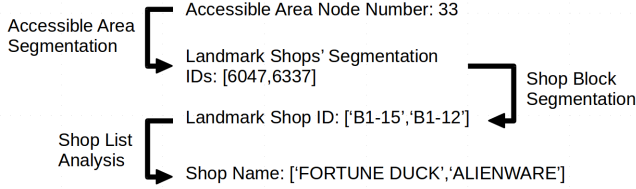
In real life, a user is interested in the name of a shop rather than its ID number on the floorplan. Therefore, it is important to map shop IDs with shop names. For this purpose, shop lists is parsed before the final map construction.

A shop list is usually a table-like document with each column being either shop names or shop ID numbers. There may be several table blocks in a shop list. DLA is required here to extract blocks that contain only one table with two columns. One of the two columns should be shop names and another column should be the shop IDs. In this way, shop names and IDs can be paired correctly. Before DLA, texts need to be extracted first. We use the method same as what we used to process a floorplan, with slight modification to the filtering parameters. This gives a text salient map without disturbance components. Recursive XY-cut algorithm [11] is then performed on the salient map. In each recursion, we dichotomize the block in X direction and cut Y direction into several parts. Finally the salient map is cut into several XY-cut blocks, with each of them being a column of shop names, shop IDs or a mixture of both.

To pair separated shop name and shop ID together, the result of XY-cut is stored as a tree, where each leaf nodes represents an XY-cut block. We perform OCR on each node and classify it as a text block, an ID block or a mixed block. We call the former two types of nodes the unpaired nodes. For each unpaired node, its unpaired siblings will be tested. If the node and its siblings are of different block types, the parser will merged them into a new paired node. See Figure 4 for an example of such merging.

Each paired block is then splitted into different lines with each line being a mixture of a shop name and a shop ID. A name-ID map is then constructed between the name and the ID on such lines.

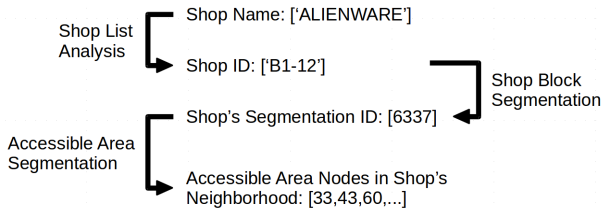
**Fig. 4.** Left: overcutted shopping instruction. Right: refined output, where each partition has only a column of shop names and a column of shop IDs.



**Fig. 5.** During map construction, each node on accessible area is mapped with a set of shop which lies in its neighborhood.

## 2.4. Map Construction

A topological map is constructed based on the topological and semantic information extracted in all steps described above. The name of a shop is mapped with its ID through analysis of shop list, and the ID of a shop is mapped with its blocks in map through analysis of floorplan. An example of map construction is shown in Figure 5. The system can thus locate itself by detecting names of shop nearby, the details are covered in next section.



**Fig. 6.** During localization process, a known shop name is mapped with a set of accessible area nodes based on constructed topological map. These nodes will then be filtered based on the subsequential image inputs.



**Fig. 7.** Collected storefront images. Some of them contain clear text logo (first row), while others may have blurred or hard-to-recognize logos (second row).

## 3. ENVIRONMENT SENSING AND LOCALIZATION

Classifying shops by detecting text logo on a shop's signboard can fail easily. The reason is multifold. Firstly, shop logo texts are usually not designed using standard fonts. Decorations or skewnesses made deliberately by designers can cause trouble to OCR softwares which are trained on standard printing fonts. Secondly, motion, luminance and visual angle variance can make logo texts hard or impossible to detect and/or correctly recognize.

Although the ngram method described in [12] has been proved to be useful for shop recognition in [13] and [14], we show in experiment that using pure ngram for shop recognition can still fail in many cases. The most severe problem with encoding a shop name using ngram is the possible ambiguity when shop logo text is incorrectly or only partly captured.

In this section, we describe a feature fusion method used to identify shops appearing in a photo. This method is inspired by the fact that many famous brands have their shops decorated in a unified style. If such style can be captured and utilized jointly with ngram feature in the recognition process, we may expect some gain in the accuracy.

### 3.1. Data Collection

We collected shop photos of 51 different shop classes from Google. These shop classes include a wide range of brands, from Adidas to Zippo. The images illustrates a view of the storefront, including the style of decoration, tone of color and optionally a shop logo. For each shop class, there are at least 20 images available. For some shops, up to 60 images are collected. We collected 2,738 photos in total, some of which are shown in Figure 7.

### 3.2. Retrieving Style Features

In order to retrieve style features from the image, we fine-tuned the AlexNet [15] using our dataset to adapt it for our

purpose. After 50,000 iterations, the test accuracy reached 46.25%. This is rather satisfying, given the fact that many brands look quite the same if no logo is observed. In the worst case, a random guess can only achieve an accuracy lower than 2%.

We then fetch the output of the 7th layer, the fc-7 layer, and use it as our style feature vector. The AlexNet takes a patch with fixed size 224\*224 as the input and our collected images usually have sizes larger than that, so we randomly pick 16 patches in an image and calculate each patch's feature vector. After that, we gain the final style feature vector by taking the maximum of each bin (we call this bin-wise max-pooling). A 4096-dimension vector is hereby retrieved to represent style feature of the image.

### 3.3. Retrieving Text Features

In order to extract text features from a image, we first need to locate text areas. In our practice, we used [16]'s method to get a text salient map. After that, we binarize the salient map, run RLSA on it and get one or more bounding boxes as text area candidates. Some candidates are rejected by geometrical features such as box width and width-height ratio.

We then run [13]'s method on filtered candidates. This method produces an  $1 \times 10000$  ngram vector predefined using some text corpus data. A softmax classifier is then trained to reject false text candidates. To train the classifier, we use 10000 ngram bins plus 4 geometrical features:  $w$ ,  $h$ ,  $w + h$  and  $w/h$ , which corresponds to the width, height, scale and shape of the box. This classifier can distinguish true boxes from false ones with an accuracy of 93.89%. Finally, the candidates classified as true positive ones are used to generate text feature vectors.

### 3.4. Combining Style and Text Features

There two ways by which we can perform feature fusion. One way is to get class score from ngram classifier and style classifier separately and add the two scores together as the final class score. This method can be formally expressed as:

$$\vec{y} = W_t \vec{x}_t + W_s \vec{x}_s, \quad (3)$$

where  $\vec{y} \in \mathbb{R}^{M \times 1}$  is a vector of class scores for one sample,  $\vec{x}_t \in \mathbb{R}^{P \times 1}$  is the ngram feature vector,  $\vec{x}_s \in \mathbb{R}^{Q \times 1}$  is the style feature vector, and  $W_t \in \mathbb{R}^{M \times P}$ ,  $W_s \in \mathbb{R}^{M \times Q}$  are learned weights for style classifier and ngram classifier, respectively. As style feature and ngram feature come from different network, some scaling is required to make each of them of same importance.

Another way is to combine ngram feature and style feature into a new feature vector and train a classifier on the joint

vectors. Formally, class scores can be calculated as:

$$\vec{y} = f_{activation}(W \vec{x}), \quad (4)$$

with  $\vec{x} \in \mathbb{R}^{(P+Q) \times 1}$  being the joint feature vector,  $\vec{y} \in \mathbb{R}^{M \times 1}$  being the class score,  $W \in \mathbb{R}^{M \times (P+Q)}$  being the learned joint feature classifier and  $f_{activation}$  being the activation function.

We did experiments on both methods and find that the latter structure performs better than the former one. Therefore, we adopt the latter as our final solution. The recognition architecture is shown in Figure 8. Experiment details are covered in **Experiments and Discussion** section.

### 3.5. Localization and Navigation

Once the shop name in real scene are recognized, it can be mapped with the block on the floorplan. Figure 6 shows how this mapping works. Because the node-shop map is a many-to-many association, it requires at least two different shop observation to locate the user in most cases. For the first shop name, nodes on AA are searched and nodes with landmarks corresponding to the shop name are added to candidate location list. This forms the initial guess of the current position. For the second shop name, operation above is repeated, but only node that is adjacent or equal to the nodes added to list in the last step is considered as candidates. Finally the user will be located at a certain node or a small set of nodes that are adjacent to each other.

A navigation process involves two operations: localization and path-planning. Both of them are easy to implement on our topological map. The method used for localization has been explained above. For path planning, the system gets user's text input for destination and search for the node related to the destination. Dijkstra's algorithm is then used to find the shortest path between the destination node and the current node. Figure 9 shows an example of path planning.

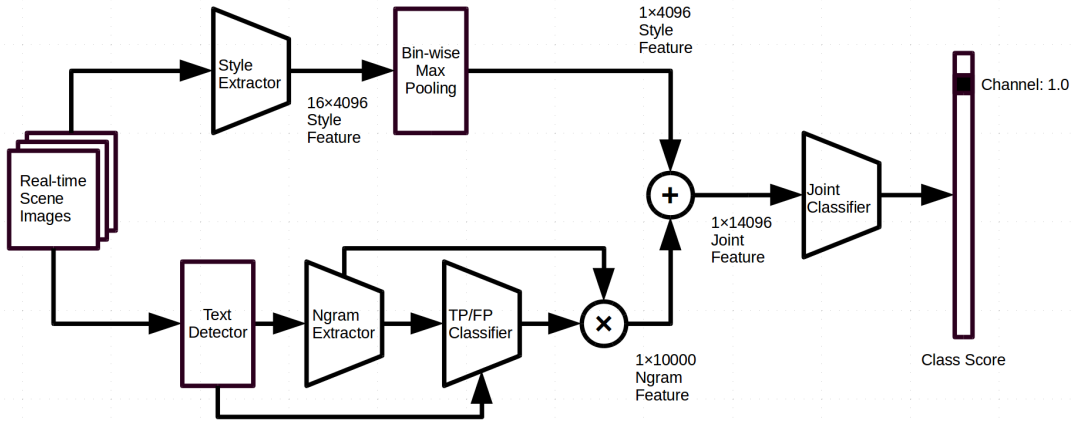
Inside a shopping mall, a user usually knows nothing about absolute directions (i.e., north, south, west and east) and there is no way to tell left and right simply from vision, so we make the navigation "shop-based". That is to say, the system prompts the user to walk towards a certain sequence of shops that lies on the path leading to the destination.

## 4. EXPERIMENTS AND DISCUSSION

### 4.1. Logistic Regression

To prove that learning-based method can achieve higher accuracy, we first compared our learning-based method with the method proposed in [14] on pure ngram feature dataset. This is equivalent to setting  $W_s$  in equation 3 to 0:

$$\vec{y} = f_{activation}(W_t \vec{x}_t), \quad (5)$$



**Fig. 8.** Environment Sensing System Architecture



**Fig. 9.** A simulated path planning from one shop to another. Dark blocks indicate the origin shop and the destination shop. Dark dots indicate different nodes on the path.

[14]’s method gets ngram vectors and compare the vector with a known list of shop names. Each ngram vector bin corresponds a combination of characters (it can be 0, 1, a, b, ab, abc, ... for details, please read Jaderberg’s arXiv paper), and the combination may or may not appear in known shop names. For each ngram vector, Wang’s method iterates over the shop name list. For each shop name, it add together all scores of combinations which appear in it to get a score for the class. Then it select the shop name with the highest class score as the prediction of the ngram vector. Therefore, [14]’s method set weights  $W_t$  in equation 5 manually as:

$$W_t(i, j) = \begin{cases} 1 & gram_j \in shop\_name_i, \\ 0 & else. \end{cases} \quad (6)$$

We, instead, use logistic regression to learn such weights  $W$ . We manually label text bounding boxes candidates as true-positive (TP) samples and false-positive (FP) ones, and use this two sets to train two logistic regression model (in practice, a single-layer neural network with sigmoid activation is used):

$$\vec{y} = f_{sigmoid}(W_{LR}\vec{x}), \quad (7)$$

where  $W_{LR}$  is the learned weight and  $f_{sigmoid}$  is the sigmoid activation function [17].

There are 2077 TP samples and 10590 FP samples in the training set, and 554 TP samples and 2653 FP in the test set. The quantitative test result is shown in Table. 1. From the table, we see the following facts:

1. Using logistic regression can achieve higher accuracy than manually setting combination weights.
2. Logistic regression has learned from TP samples and tries to avoid fitting the FP samples. This can be known from the fact that both LR1 and LR2 model performed



MODEL	TRAINING	TEST	ACCURACY(%)
LOST_S	-	TP	71.30
LOST_S	-	FP	2.86
LOST_S	-	AVG	15.15
LR1	NGRAM-TP	TP	83.94
LR1	NGRAM-TP	FP	3.20
LR1	NGRAM-TP	AVG	17.15
LR2	NGRAM-TP+FP	TP	74.01
LR2	NGRAM-TP+FP	FP	5.54
LR2	NGRAM-TP+FP	AVG	17.37

**Table 1.** Training and testing classifiers on TP/FP dataset. The AVG results are weighted average of TP/FP accuracy.

poorly on FP test samples but fairly well on TP samples. This indicates that filtering FP samples before testing can improve the accuracy.

## 4.2. Feature Fusion

We train the logistic regression model LR3A (letter "A" means "additive") using classified TP ngram samples and max-pooled style features. In test phase, we gain test scores following equation 3.

Since we would prefer a data-driven way to simply adding two scores together, we trained model LR3. To train this model, we combine style feature and classified TP text feature vector together to form a  $1 \times 14096$  feature vector. Class scores are calculated using equation 4, with activation function specified as sigmoid.

The test set is constructed in the same way. The final test result is shown in Table 2. To make it fair, we test all ngram-based models using the same classified TP ngram feature (NGRAM-CTP). From Table 2, we know that feature fusion model LR3A and LR3 outperforms methods proposed in [14] on our dataset. Comparing model STYLE\_ONLY, LR2 and LR3, which was trained on merely style feature, merely text feature, and mixed text-style feature respectively, we draw the conclusion that joint feature can represent a shop better than separate ones.

Figure 10 shows some cases in our experiments. The first column (the "adidas" example) shows how ngram ambiguity affect the result. The ngram correctly capture some combination like "a" and "ad" but failed to capture "di" because "d" and "i" are too close to each other, leading to a wrong prediction "prada". The second column (the "Calvin Klein" example) shows yet another example of ngram ambiguity. In the third example (the "zara" example), word "zara" is completely flooded by false text saliency information generated by [16]'s method. In all three cases, where text detection failed, feature fusion method could correctly make the right decision.

However, there are cases when feature fusion fails. In the last example (the "Tiffany" example), ngram method was misguided by the tailing letter "o". The feature fusion method was also misled by the blue adornments and clothes, leading

MODEL	TRAINING SET	TEST SET	ACCURACY(%)
LOST_S	-	NGRAM-CTP	66.31
STYLE_ONLY	STYLE	STYLE	62.92
LR3A	NGRAM-TP+STYLE	NGRAM-CTP+STYLE	82.00
LR3	NGRAM-CTP+STYLE	NGRAM-CTP+STYLE	86.27

**Table 2.** Comparison between [14]'s method, style-only model, additive model and joint trained model.

to a false "Fanc1" output (Fanc1's storefronts are mostly blue as shown in the last column).

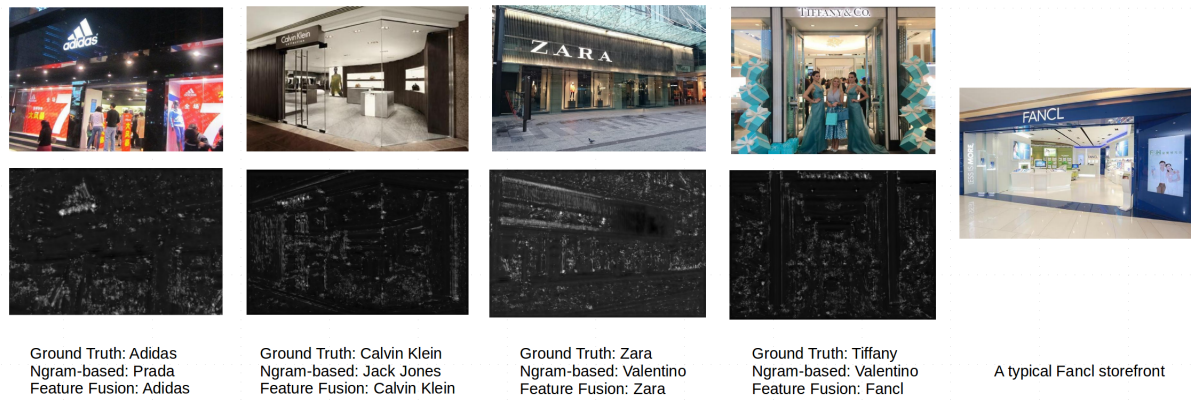
## 5. CONCLUSION

In this article, we proposed an indoor positioning system that works in shopping malls. We put forward an automatic way of interpreting shopping mall floorplans and shop lists. Moreover, a feature fusion method is introduced to recognize shops. This method utilizes both text feature extracted as ngram vectors and style feature extracted from a fine-tuned AlexNet. We showed by experiments that feature fusion can improve the accuracy of shop recognition. However, as there's no large-scale storefront dataset publically available, we are not able to compare our method with others' thoroughly at present.

While we see a great performance improvement by introducing feature fusion method, there are still some cases where feature fusion failed. The main problem with the current method is that both ngram feature and style feature are not robust enough for disambiguating a shop class. We will look into this problem in future works. We plan to continue this work by building an application system and test it in real shopping circumstances. Building a standard benchmark dataset is also a possible future work.

## 6. REFERENCES

- [1] Kevin Curran, Eoghan Furey, Tom Lunney, Jose Santos, Derek Woods, and Aiden McCaughey, "An evaluation of indoor location determination technologies," *J. Locat. Based Serv.*, vol. 5, no. 2, pp. 61–78, June 2011.
- [2] Ahmed Wasif Reza and Tan Kim Geok, "Investigation of indoor location sensing via rfid reader network utilizing grid covering algorithm," *Wireless Personal Communications*, vol. 49, no. 1, pp. 67–80, 2009.
- [3] Xiaohan Liu, Hideo Makino, and Kenichi Mase, "Improved indoor location estimation using fluorescent light communication system with a nine-channel receiver," *IEICE Transactions*, vol. 93-B, no. 11, pp. 2936–2944, 2010.
- [4] N. Chang, R. Rashidzadeh, and M. Ahmadi, "Robust indoor positioning using differential wi-fi access points," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1860–1867, Aug 2010.



**Fig. 10.** Visualization of some experiments. The first row is the original image. The second row is the text saliency from which text bounding boxes are generated. The third row is the ground-truth information and outputs of ngram-based method and our feature fusion method.

- [5] David Obdržálek, Stanislav Basovník, Lukáš Mach, and Andrej Mikulík, *Detecting Scene Elements Using Maximally Stable Colour Regions*, pp. 107–115, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [6] Y. Li and H. Lu, “Scene text detection via stroke width,” in *Pattern Recognition (ICPR), 2012 21st International Conference on*, Nov 2012, pp. 681–684.
- [7] Ray Smith, “An overview of the tesseract ocr engine,” in *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 629–633.
- [8] Damien Garcia, “Robust smoothing of gridded data in one and higher dimensions with missing values,” *Computational Statistics & Data Analysis*, vol. 54, no. 4, pp. 1167 – 1178, 2010.
- [9] Guojie Wang, Damien Garcia, Yi Liu, Richard de Jeu, and A. Johannes Dolman, “A three-dimensional gap filling method for large geophysical datasets: Application to global satellite soil moisture observations,” *Environmental Modelling & Software*, vol. 30, pp. 139 – 142, 2012.
- [10] Charles R Dyer, “Computing the euler number of an image from its quadtree,” *Computer Graphics and Image Processing*, vol. 13, no. 3, pp. 270 – 276, 1980.
- [11] Jaekyu Ha, R. M. Haralick, and I. T. Phillips, “Recursive x-y cut using bounding boxes of connected components,” in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2) - Volume 2*, Washington, DC, USA, 1995, ICDAR ’95, pp. 952–, IEEE Computer Society.
- [12] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet, “Multi-digit number recognition from street view imagery using deep convolutional neural networks,” 2013.
- [13] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, “Synthetic data and artificial neural networks for natural scene text recognition,” 2014.
- [14] Raquel Urtasun Shenlong Wang, Sanja Filder, “Lost shopping! monocular localization in large indoor spaces,” in *ICCV*, 2015.
- [15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.
- [16] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Deep features for text spotting,” in *European Conference on Computer Vision*, 2014.
- [17] Jun Han and Claudio Moraga, *The influence of the sigmoid function parameters on the speed of backpropagation learning*, pp. 195–201, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.